
X1 Porting Experiences
Tom Baring
Arctic Region Supercomputing Center

X1 Porting Experiences

Every **code ported to the X1 should be analyzed for performance...**

X1 Porting Experiences

Outline

- Six case studies
- Conclusions

BH

- **Ab initio quantum chemistry code for predicting material properties**

Language	Fortran / C
Explicit Parallelism	MPI
Lines of code	~9,500

* X1 Port of BH in collaboration with ERDC MSRC

BH Porting Issues

- **Complexity of build process and code**
- **Data size problems:**
  - **`-s default64`** required
  - However, some library routines require 32-bit arguments
    - ERFC (PE 5.0 and earlier)
    - FLUSH
- **Missing utilities:**
  - replace **`itime`** with **`date_and_time`**

BH Optimization

- **Initial `cray_pat` profile**
  - **scratch file I/O at top**
- **Optimization: use FFIO to map scratch files to memory:**

```
export FILENV=\$EVAR
eval $(assign -F mr.scr.ovfl.start_size=72  p:temp%)
eval $(assign -F mr.scr.ovfl.start_size=10342 p:WAVE%)
```

BH Optimization

- **excno** subroutine. One loop, 100 lines...
  - Inlined subroutines... another 150 lines...
  - Dependence on **ecrs**:

```
ecrs = -1     !!! initialize
do 8100 n=1,nplwv
    if (igga.GT.0) then        !!! igga is loop invariant.
      ecrs = …
    else
      !!! no reference to ecrs
    endif
    if (igga.EQ.1) then
      … = ecrs …
    endif
8100 continue
```

BH Optimization

- potnl subroutine:

```
386.   1 2 M 4---------<          do m=1, nallkp(n,iks)
387.   1 2 M 4                      cfcf=catstf(m,l,j,n)*flqint(m,ml,j,n)*cg(m)
388.   1 2 M 4
389.   1 2 M 4 Vs------<            do jj1=1,3
390.   1 2 M 4 Vs W----<              do jj2=1,3
391.   1 2 M 4 Vs W W--<                do jj3=1,3
392.   1 2 M 4 Vs W W                     ctt(jj1,jj2,jj3)=
393.   1 2 M 4 Vs W W          $          ctt(jj1,jj2,jj3)+rleng(jj1,m,n)
394.   1 2 M 4 Vs W W          $          *rleng(jj2,m,n)*rleng(jj3,m,n)*cfcf
395.   1 2 M 4 Vs W W-->                enddo
396.   1 2 M 4 Vs W---->              enddo
397.   1 2 M 4 Vs------>            enddo

Recurrence on ctt, line 392
```

BH Optimization

- potnl:
  - Cray ftn directives, and manually unwind

```
 382.  1 2 3           !DIR$ PREFERSTREAM
383.  1 2 3           !DIR$ PREFERVECTOR
384.  1 2 3 MV------<  do m=1, nallkp(n,iks)
385.  1 2 3 MV            cfcf=catstf(m,l,j,n)*flqint(m,ml,j,n)*cg(m)
386.  1 2 3 MV
399.  1 2 3 MV      ctt(1,1,1)=ctt(1,1,1)+rleng(1,m,n)*rleng(1,m,n)*rleng(1,m,n)*cfcf
400.  1 2 3 MV      ctt(1,1,2)=ctt(1,1,2)+rleng(1,m,n)*rleng(1,m,n)*rleng(2,m,n)*cfcf
401.  1 2 3 MV      ctt(1,1,3)=ctt(1,1,3)+rleng(1,m,n)*rleng(1,m,n)*rleng(3,m,n)*cfcf
402.  1 2 3 MV      ctt(1,2,1)=ctt(1,2,1)+rleng(1,m,n)*rleng(2,m,n)*rleng(1,m,n)*cfcf
403.  1 2 3 MV      ctt(1,2,2)=ctt(1,2,2)+rleng(1,m,n)*rleng(2,m,n)*rleng(2,m,n)*cfcf
...
```

BH Optimization

- forcnl:

```
500.   1 2 W 4 5----<              do 9401 ml = 1,nl(j)
501.   1 2 W 4 5                     if (ilall(ml,j).eq.0) then
502.   1 2 W 4 5 Vr-<                  do 2010 m=2,nallkp(n,iks)
503.   1 2 W 4 5 Vr                      cpart(m)=catstf(m,l,j,n)
504.   1 2 W 4 5 Vr       $                   *cptwfp(m,nn,n)
505.   1 2 W 4 5 Vr       &                   *flqint(m,ml,j,n)
506.   1 2 W 4 5 Vr                      ct1=ct1+cpart(m)*lpctfx(igx(m,n))
507.   1 2 W 4 5 Vr                      ct2=ct2+cpart(m)*lpctfy(igy(m,n))
508.   1 2 W 4 5 Vr                      ct3=ct3+cpart(m)*lpctfz(igz(m,n))
509.   1 2 W 4 5 Vr->  2010            continue
510.   1 2 W 4 5                     endif
511.   1 2 W 4 5                     if (ilall(ml,j).eq.1) then
512.   1 2 W 4 5 V--<                  do 2011 m=2,nallkp(n,iks)
513.   1 2 W 4 5 V                       cpart(m)=catstf(m,l,j,n)
```

BH Optimization

- forcnl:
  - eliminate conditionals and recurrence with temp arrays

```
534.   1 2 W 4 MV----<                    do m=2, nallkp(n,iks)
535.   1 2 W 4 MV r--<                      do ml = 1, nl(j)
536.   1 2 W 4 MV r                          cpart2(m,ml)=catstf(m,l,j,n)*
537.   1 2 W 4 MV r          $                   cptwfp(m,nn,n)*flqint(m,ml,j,n)
538.   1 2 W 4 MV r-->                      enddo
539.   1 2 W 4 MV---->                    enddo
540.   1 2 W 4 MV----<                    do 2010 m=2,nallkp(n,iks)
541.   1 2 W 4 MV 6--<                      do ml=1, numilall0
542.   1 2 W 4 MV 6                          ct1=ct1+cpart2(m,ilallis0(ml))*
543.   1 2 W 4 MV 6          $                   lpctfx(igx(m,n))
```

BH Optimization

- forcnl (farther down):

```
907.  1 2 W 4 M 6 7-----------<          do i=1,3
908.  1 2 W 4 M 6 7 8----------<             do i2=1,3
909.  1 2 W 4 M 6 7 8 9--------<                do i3=1,3
910.  1 2 W 4 M 6 7 8 9 W------<                   do m=1,3
911.  1 2 W 4 M 6 7 8 9 W W----<                      do m2=1,3
912.  1 2 W 4 M 6 7 8 9 W W W--<                         do m3=1,3
913.  1 2 W 4 M 6 7 8 9 W W W                              cterm=cterm+
914.  1 2 W 4 M 6 7 8 9 W W W      &                         ceinl3(i,i2,i3,k)
915.  1 2 W 4 M 6 7 8 9 W W W      $                         *conjg(cg3ank(
916.  1 2 W 4 M 6 7 8 9 W W W      $                          l,j,m,m2,m3,nn,n))
917.  1 2 W 4 M 6 7 8 9 W W W      $                         *amet(i,m)*
918.  1 2 W 4 M 6 7 8 9 W W W      $                          amet(i2,m2)*
919.  1 2 W 4 M 6 7 8 9 W W W      $                          amet(i3,m3)
920.  1 2 W 4 M 6 7 8 9 W W W-->                         enddo
921.  1 2 W 4 M 6 7 8 9 W W---->                      enddo
922.  1 2 W 4 M 6 7 8 9 W------>                   enddo
923.  1 2 W 4 M 6 7 8 9-------->                enddo
924.  1 2 W 4 M 6 7 8---------->             enddo
925.  1 2 W 4 M 6 7----------->          enddo
```

BH Optimization

- forcnl (farther down):
  - Eliminate loop. Streaming replaced by vectorization:

```
964.  1 2 W 4 V i---------------<          do i=1,3
965.  1 2 W 4 V i i-------------<             do i2=1,3
966.  1 2 W 4 V i i ir----------<                do i3=1,3
967.  1 2 W 4 V i i ir 9--------<                   do m=1,3
968.  1 2 W 4 V i i ir 9 10-----<                      do m2=1,3
969.  1 2 W 4 V i i ir 9 10 11--<                         do m3=1,3
970.  1 2 W 4 V i i ir 9 10 11                              cterm=cterm+
971.  1 2 W 4 V i i ir 9 10 11        &                          ceinl3(i,i2,i3,k)
972.  1 2 W 4 V i i ir 9 10 11        $                          *conjg(cg3ank(
973.  1 2 W 4 V i i ir 9 10 11        $                          l,j,m,m2,m3,nn,n))
974.  1 2 W 4 V i i ir 9 10 11        $                          *amet(i,m)*
975.  1 2 W 4 V i i ir 9 10 11        $                          amet(i2,m2)*
976.  1 2 W 4 V i i ir 9 10 11        $                             amet(i3,m3)
977.  1 2 W 4 V i i ir 9 10 11-->                            enddo
978.  1 2 W 4 V i i ir 9 10----->                         enddo
979.  1 2 W 4 V i i ir 9-------->                      enddo
980.  1 2 W 4 V i i ir---------->                   enddo
981.  1 2 W 4 V i i------------->                enddo
982.  1 2 W 4 V i--------------->             enddo
```

BH Performance

- "BIG" Benchmark run time, hours

Code Version	Memory Mapped	X1 4 MSPs	SX-6 4 CPUs
Original		2.8	4.6
Vectorized		2.5	3.3
Original	√	1.7	-
Vectorized	√	1.1	-

FLAPW / PFLAPW

- **Ab initio quantum chemistry codes for predicting material properties**

Program	FLAPW	PFLAPW
Language	Fortran	Fortran
Explicit Parallelism	none	MPI / ScaLAPACK
Lines of code	~47,000	~58,000

FLAPW / PFLAPW Porting Issues

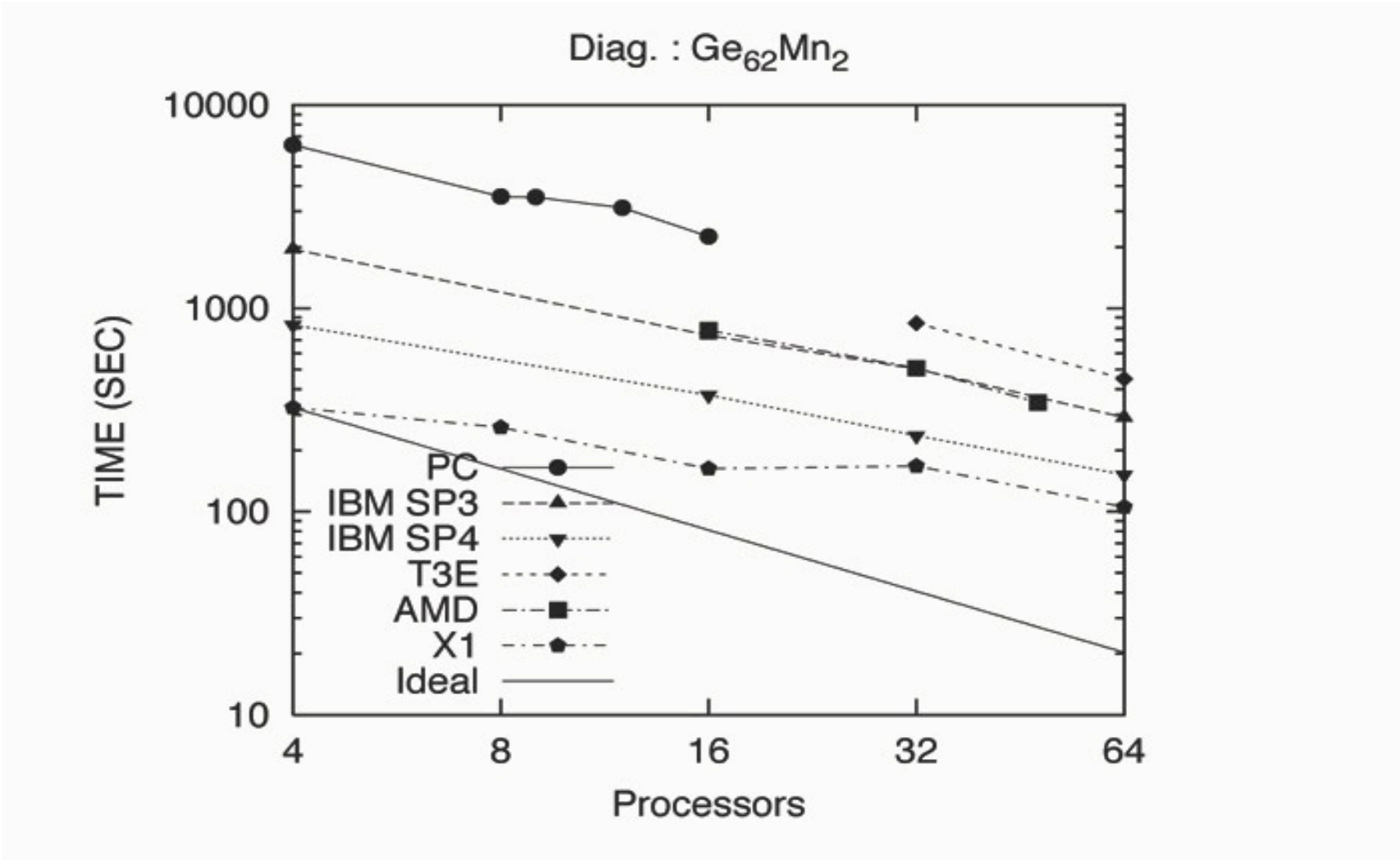
- **FLAPW:**
  - no problems.
    - **`-s default64`** just worked...
- **PFLAPW:**
  - library support
    - **`-s default64`** required
    - no **`default64`** version of BLACS / ScaLAPACK

PFLAPW Porting Issues

- **Custom Fortran pre-processor replaces -s default64 :**

Translate From:	To:
REAL declarations	DOUBLE PRECISION
COMPLEX declarations	COMPLEX*16
IMPLICIT REAL (A-H,O-Z)	IMPLICIT DOUBLE PRECISION (A-H,O-Z)
(files lacking IMPLICIT)	IMPLICIT DOUBLE PRECISION (A-H,O-Z)
Real literals	KIND=8
REAL, CMPLX, SIGN	DREAL, DCMPLX, DSIGN
ALOG, AMAX1, AMIN1	LOG, MAX, MIN

PFLAPW Performance



Gasoline

- **Galaxy formation model: N-body particle simulation, tree based gravity code, hydrodynamics extensions.**

Language	C
Explicit Parallelism	MPI / SHMEM
Lines of code	~38,000

Gasoline Porting Issues

- **None**

Gasoline Optimization

- **SSP Mode (~ 4x speedup over MSP)**

Gasoline Optimization

- **Costliest loops (7 iterations per loop):**

```
 54.  1 2------<    for (ix=-nEwReps;ix<=nEwReps;++ix) {
 55.  1 2             bInHolex = (ix >= -nReps && ix <= nReps);
 56.  1 2             dxo = dx + ix*L;
 57.  1 2 3----<      for (iy=-nEwReps;iy<=nEwReps;++iy) {
 58.  1 2 3             bInHolexy = (bInHolex && iy >= -nReps && iy <= nReps);
 59.  1 2 3             dyo = dy + iy*L;
 60.  1 2 3 4--<        for (iz=-nEwReps;iz<=nEwReps;++iz) {
 61.  1 2 3 4             bInHole = (bInHolexy && iz >= -nReps && iz <= nReps);
 [...]
100.  1 2 3 4             if (bInHole) gam[0] = -erf(alpha*r);
101.  1 2 3 4             else gam[0] = erfc(alpha*r);
```

- **erf / erfc inlined**
  - inner loop successfully vectorized
  - no speedup

Gasoline Optimization

- **Loop nest manually collapsed:**

```
241.  1            ixStride = (2*nEwReps + 1)*(2*nEwReps + 1);
242.  1            iyStride = 2*nEwReps + 1;
243.  1 V----<     for (tmp_n = 0; tmp_n < ixStride*iyStride; tmp_n++) {
244.  1 V             ix = tmp_n/ixStride - nEwReps;
245.  1 V             iy = (tmp_n%ixStride)/iyStride - nEwReps;
246.  1 V             iz = tmp_n%iyStride - nEwReps;
247.  1 V             bInHolex = (ix >= -nReps && ix <= nReps);
248.  1 V             dxo = dx + ix*L;
249.  1 V             bInHolexy = (bInHolex && iy >= -nReps && iy <= nReps);
250.  1 V             dyo = dy + iy*L;
```

- **Speedup:**
  - 3x for subroutine
  - 50% overall

Gasoline Performance

Platform	GFLOPS	nprocs	% of peak
X1	4.0	16 SSPs	7.8
T3E (450mhz)	4.6	64 PEs	7.9

Rosetta

- **Ab initio protein structure prediction**

Language	Fortran
Explicit Parallelism	none
Lines of code	~43,300

Rosetta Porting Issues

- **Missing Fortran API to `SYSTEM` call**
  - Wrote C interface

Rosetta Optimization

- **cray_pat reports 45% of time in `count_pair_position`**

```
358.   1 2 3----------------<          do jatom=1,5
359.   1 2 3                             type1=Eatom_type(jatom,j)
360.   1 2 3                             xpos1 = Eposition(1,jatom,j)
361.   1 2 3                             ypos1 = Eposition(2,jatom,j)
362.   1 2 3                             zpos1 = Eposition(3,jatom,j)
363.   1 2 3
364.   1 2 3                             if (type1.ne.0) then
367.   1 2 3 4--------------<              do iatom=1,5
368.   1 2 3 4                               type2=Eatom_type(iatom,i)
369.   1 2 3 4 V I I I I I-<>                if (count_pair_position(i,type2,res(i),
370.   1 2 3 4                    &                        j,type1,res(j))) then
```

Rosetta Performance

- **Timings:**
  - Power4 : 74 seconds
  - X1 : 30 minutes
- **Decided against further effort**

FREEH

- **Superconduction in ferromagnetic materials**

Language	Fortran 95
Explicit Parallelism	none
Lines of code	255

FREEH Porting Issues

- **none**

FREEH Optimization and Performance

- **Optimization:**
  - 95% of time in LAPACK routine, SSPEV.
  - No optimization attempted.
- **Performance:**
  - (Reported by the user):
    - Power4 1.3x faster than X1

Dyson

- **Electrodynamics studies of photonic crystals**

Language	Fortran
Explicit Parallelism	none
Lines of code	290

Dyson Porting Issues

- **none**

Dyson Optimization

- **Original version: 37 MFLOPS**
- **Inspection pointed to these loops:**

```
191.   1 Vp----<   do i = 1, nxy ; do j = 1, n_epsilon
192.   1 Vp          if(i /= km(order) .and. j /= order) then
193.   1 Vp              g(i, j) = g(i, j) + wcda*g(i,order)        &
                                 *eps(order)*g(km(order), j)
194.   1 Vp          endif
195.   1 Vp          if(i == km(order) .and. j == order) then
196.   1 Vp              g(i, j) = (z1/wcda)/(1. - z1*eps(order))
197.   1 Vp          endif
198.   1 Vp---->   enddo ; enddo
```

```
A loop starting at line 191 was conditionally vectorized.
A loop starting at line 191 was not multi-streamed because a recurrence
   was found on "G" at line 193.
```

Dyson Optimization

- **Rewritten with temp arrays: 3820 MFLOPS**

```
206. 1 Vw V M-<>  G_KM_ORDER_J(:) = g(km(order),:)   ! Save
207. 1 V M---<>   G_I_ORDER(:) = g(:,order)          ! Save
208. 1
209. 1            ! Almost all the work in the program is in this loop.
210. 1 Vm------<  do i = 1, nxy
211. 1 Vm Mr---<    do j = 1, n_epsilon
212. 1 Vm Mr          g(i, j) = g(i, j) + wcda                          &
                              *G_I_ORDER(i)*eps(order)*G_KM_ORDER_J(j)
213. 1 Vm Mr--->    enddo
214. 1 Vm------>  enddo
215. 1
216. 1 V M---<>   g(:, order) = G_I_ORDER(:)          ! Restore
217. 1 Vw V M-<>  g(km(order), :) = G_KM_ORDER_J(:)   ! Restore
```

Dyson Optimization

- **`INTERCHANGE` compiler directive: 3378 MFLOPS**

```
210.  1              !DIR$ INTERCHANGE (j,i)
211.  1 iV-----<     do i = 1, nxy
212.  1 iV iM--<       do j = 1, n_epsilon
213.  1 iV iM            g(i, j) = g(i, j) + wcda                          &
                                  *G_I_ORDER(i)*eps(order)*G_KM_ORDER_J(j)
214.  1 iV iM-->       enddo
215.  1 iV----->     enddo
```

- **Manually interchanged: 4223 MFLOPS**

```
211.  1 Mr-----<     do j = 1, n_epsilon
212.  1 Mr V---<       do i = 1, nxy
213.  1 Mr V             g(i, j) = g(i, j) + wcda                          &
                                  *G_I_ORDER(i)*eps(order)*G_KM_ORDER_J(j)
214.  1 Mr V--->       enddo
215.  1 Mr----->     enddo
```

Dyson Performance

- User problem
- Run time, seconds

Version	X1	SX-6	Power4
original	43501	21773	833
vectorized	161	329	822

X1 Porting Experiences

- **Conclusions**
  - Porting isn't too difficult.
  - X1 is not a general purpose machine. Codes must vectorize.
  - ***Every*** code ported to the X1 should be analyzed for performance...
    - minor tuning of good code can provide significant speedup
    - a code should not be rejected for the X1 without some analysis

This work was supported in part by a grant of HPC resources from the Arctic Region Supercomputing Center at the University of Alaska Fairbanks as part of the Department of Defense High Performance Computing Modernization Program.